Documentation of Condor at USGS CIDA-EMU

By Vladimir Brik and Michael N. Fienen

Open-File Report 2012–xxxx

U.S. Department of the Interior
U.S. Geological Survey

U.S. Department of the Interior
KEN SALAZAR, Secretary

U.S. Geological Survey
Marcia K. McNutt, Director

U.S. Geological Survey, Reston, Virginia 2012

For product and ordering information:
World Wide Web: http://www.usgs.gov/pubprod
Telephone: 1-888-ASK-USGS

For more information on the USGS—the Federal source for science about the Earth, its natural and living resources, natural hazards, and the environment:
World Wide Web: http://www.usgs.gov
Telephone: 1-888-ASK-USGS

Suggested citation:
Brik, V. and Fienen, M. N., 2012, Documentation of Condor at USGS CIDA-EMU: U.S. Geological Survey Open-File Report 2012–xxxx, xx p. [http://pubs.usgs.gov/of/2011/xxxx]

Any use of trade, product, or firm names in this publication is for descriptive purposes only and does not imply endorsement by the U.S. Government.

Typeset on October 26, 2011

Contents

Figures

Blank page

Documentation of Condor at USGS CIDA-EMU

By Vladimir Brik and Michael N. Fienen

Overview

This document illustrates a Condor framework for running at the USGS CIDA–EMU (Center for Integrated Data Analytics–Environmental Modeling Unit). Two example applications are presented; one using beoPEST (**?**) and another implementing a leave-one-out-cross validation study (**?**) .

The first example, beoPEST, is a master-worker code performing model-independent parameter estimation using the techniques of PEST (**?**). beoPEST is used to distribute the embarrassingly parallel operation of calculating a Jacobian matrix of observation to parameter sensitivities. This operation requires multiple independent model runs, coordinated by a master. The master opens a TCP-IP port on which to listen for available workers. beoPEST is fault-tolerant so workers are allowed to come online and go offline. This configuration is amenable to implementation on Condor. The role of Condor is to start a persistent master and form a queue of potential workers. The master must be exempt from eviction, but workers may be evicted, in which case Condor will start another worker once resources become available. The exemption from eviction for the master can be accomplished either by running on the same machine from which the entire Condor job is submitted, or by specifying a particular target machine on which eviction will not take place. Examples of both processes are presented in this document.

The second example, using leave-one-out cross validation, shows a more general approach to using Condor for a problem in which a problem must be run a large number of times in parallel. A similar approach could be easily applied to a Monte Carlo analysis. In this case, Python code is used to run particular sets of parameters.

Files for Windows beoPEST Run with Local Master

In this example, Python is required to be deployed in addition to the other typically needed files. If Python is not required (or is already deployed on the worker nodes) then ”Python27.zip” can be removed from the line of `mw.sub` that starts with `transfer_input_files=` and the first two lines of The executable in this case is a windows batch file called `worker.bat`.

To run this problem, the user must configure the two files `mw.sub` and `worker.bat` (both described below). Then, on the command line, type “`condor_submit mw.sub`”. Log files including Condor logs, and both standard out and standard error from beoPEST are available in the `condor_output` subdirectory. The command `condor_q` highlights the activity of the current user while `condor_status` gives a summary of the overall network activity (including which resources are available and which are in use). Finally, to stop a job after it is submitted, type “`condor_rm -all`”. This removes all jobs by a current user. Alternatively, single jobs can be killed. The job is refered to as the cluster (and can be identified from `condor_q` output). So, to kill cluster 100, type “`condor_rm 100`”.

Condor Submit File to Start Master and Workers: `mw.sub`

In the following, some preparation is required.

1. In the master folder, a subfolder called `condor_output` must be created.

2. All files needed by beoPEST in slave mode, including PST files, all executables, and all model files, must be zipped into a file called `WORKER_DATA.zip` that resides in the master folder. This folder should initially have been named “data” such that, upon unzipping, a folder called `data` is created. This can be changed if accompanied by changes to `worker.bat`.

In the listings below, all text surrounded by angled braces (<·>) indicates variables that should be replaced with their values in the file. The following are defined:

`<casename>` The root of the PEST run, such that the control file is `<casename>`.pst.

`<master port>` The port on the master machine through which beoPEST master will communicate with slaves.

`<requested number of nodes>` This is the number of nodes requested. Note that if not enough machines meeting the requirements specified elsewhere in this submit file are available, fewer machines will be used.

```
#
#Condor Windows submit file for beoPEST with Master and Workers
#
notification = Never
#################################################################
# Start the master on the local machine
#################################################################
universe = local
log = condor_output/<casename>_$(Cluster).log
output = condor_output/<casename>_$(Cluster).out
error = condor_output/<casename>_$(Cluster).err
executable = beopest64.exe
arguments = <casename> /h :<master port>
queue
#################################################################
# Start the workers
#################################################################
universe = vanilla
PoolName==”CIDA”
log = condor_output/w_$(Cluster).log
output = condor_output/w_$(Cluster)_$(Process).out
error = condor_output/w_$(Cluster)_$(Process).err
executable = worker.bat
arguments = <casename> $ENV(HOSTNAME) <master port>
requirements = ((Target.OpSys==”WINNT61”) && (Target.Arch==”INTEL”)
        && (PoolName==”CIDA”) )
 should_transfer_files = YES
```

```
when_to_transfer_output = ON_EXIT_OR_EVICT
transfer_input_files = unzip.exe,Python27.zip,WORKER_DATA.zip
queue <requested number of nodes>
```

Executable to Start Worker: `worker.bat`

```
unzip Python27.zip
set path=%cd%\Python27;%cd%;%cd\data;%path%

unzip WORKER_DATA.zip
cd data

beopest64.exe %1.pst /h %2:%3
```

Files for Linux beoPEST Run

This section presents the raw files being used to run beoPEST in the Condor environment.

Makefile to Prepare Upload and Submit Condor Job: `Makefile`

A Makefile is used to take advantage of the ability of Make to check status of compressed directories. When Make is called with the argument `submit`, it checks the status of the data and bin folders relative to their compressed versions; if either folder has been changed since the compression was last performed, the compression is performed again and then the two condor jobs are submitted.

```
upload: upload/data.tar.gz upload/bin.tar.gz

upload/data.tar.gz: data/*
        tar czf upload/data.tar.gz data

upload/bin.tar.gz: bin/*
        tar czf upload/bin.tar.gz bin

submit: upload
        -mv log/[^_]* log/_archive
        condor_submit start_master.sub
        condor_submit start_worker.sub
        condor_q
```

Condor Submit File to Start Master `start_master.sub`

```
universe = vanilla
log = log/master_$(Cluster).log
output = log/master_$(Cluster)_$(Process).out
error = log/master_$(Cluster)_$(Process).err
should_transfer_files = YES
requirements = ((Target.Memory>=7000) && (Target.Machine == ”a1.hpc”)
                && (PoolName==”CIDA”))
request_memory = 7000
executable = job.sh
arguments = master <master machine name> <master port> <casename>
stream_output = True
stream_error = True
transfer_output_files = data/<outputfile1>
transfer_output_remaps = ”<outputfile1> = results/<outputfile1>”
when_to_transfer_output = ON_EXIT
transfer_input_files = upload/data.tar.gz, upload/bin.tar.gz
queue
```

Submit File to Start Worker `start_worker.sub`

```
notification = Never

universe = vanilla
log = log/worker_$(Cluster).log
output = log/worker_$(Cluster)_$(Process).out
error = log/worker_$(Cluster)_$(Process).err
executable = job.sh
arguments = worker <master machine name> <master port> <casename>
requirements = ((Target.Memory >= 7000) && (PoolName==”CIDA”))
request_memory = 7000
should_transfer_files = YES
when_to_transfer_output = ON_EXIT
transfer_input_files = upload/data.tar.gz, upload/bin.tar.gz
queue <40>
```

Executable to Start Master or Worker: `job.sh`

```
#!/bin/sh

role=$1
host=$2
port=$3
casename=$4

export LD_LIBRARY_PATH=$(pwd)/bin/lib:
export PATH=$PATH:$(pwd)/bin

tar xzf data.tar.gz
tar xzf bin.tar.gz
#rm data.tar.gz bin.tar.gz
cd data

if [ $role == ”worker” ]; then
        ppest $casename.pst /h $host:$port
elif [ $role == ”master” ]; then
        ppest $casename.pst /h :$port
else
        echo ”Invalid role <$role>” > /dev/stderr
fi
```

Files for Leave One Out Cross Validation

The following files were used to validate a statistical model (**?**) using Condor and Python.

Makefile to Prepare Upload and Submit Condor Job: `Makefile`

```
upload: upload/LOO_DATA.tgz

```

```
upload/LOO_DATA.tgz: LOO_DATA/*
        tar czf upload/LOO_DATA.tgz LOO_DATA

submit: upload
        -mkdir -p log/_archive
        -mv log/[^_]* log/_archive
        condor_submit LOO.sub
        condor_q
```

Condor Submit File for Leave one out Cross Validation: `LOO.sub`

```
notification = Never
universe = vanilla
log = log/loo_$(Cluster).log
output = log/loo_$(Cluster)_$(Process).out
error = log/loo_$(Cluster)_$(Process).err
should_transfer_files = YES
requirements = ( (OpSys == ”LINUX”) && (Arch == ”X86_64”))
executable = worker_loo.sh
request_cpus = 1
arguments = $(Process)
should_transfer_files = yes
when_to_transfer_output = ON_EXIT
transfer_input_files = upload/LOO_DATA.tgz
transfer_output_files = LOO_DATA/data_$(Process).dat
queue 99290
```

Executable to Start Leave-one-out Process on Each Condor Node: `worker_loo.sh`

```
#!/bin/sh

currLooInd=$1

tar xzf LOO_DATA.tgz
rm LOO_DATA.tgz

export LD_LIBRARY_PATH=.
cd LOO_DATA

./LOO_single.py $currLooInd
```

Condor Submit File for Leave one out Cross Validation: `LOO_single.py`

```
#!/usr/bin/python
import numpy as np
import sys
from LOO import LOO_Hg

#################################################################################
```

```
# Main Function
######################################################################################

Masterfile = 'NatfishFinalAllobs_20110617_MNF.csv'
Connectionsfile = 'comparable_calcs_20110630.dat'
IDS_file = 'all_IDS.dat'
allIDS = np.genfromtxt(IDS_file,dtype=int)
ind_to_drop = int(sys.argv[1])
ID_to_drop = allIDS[ind_to_drop]
del allIDS

cHg,obsHg = LOO_Hg(ID_to_drop,Masterfile,Connectionsfile)

set1 = open('summaryRESULTS.dat','r').readlines()

ofp = open('data_{0:d}.dat'.format(ind_to_drop),'w')
ofp.write('Dropped_ID--> {0}\n'.format(ID_to_drop))
for line in set1:
    ofp.write(line)
ofp.write('%20s%20s%20s\n' %('dropped_ID','modHg','measuredHg'))
ofp.write('%20d%20.8f%20.8f\n' %(ID_to_drop,cHg[0],obsHg[0]))
ofp.close()
```

References Cited

Brigham, M. E., Fienen, M. N., Donato, D. I., Wente, S. P., Lorenz, D. L., Trombley, M. M., Sanocki, C. A., 2011, Application and validation of the National Descriptive Model of Mercury in Fish (NDMMF), 10th International Conference on Mercury as a Global Pollutant, Halifax, Nova Scotia, Canada, July 25-29, 2011.

Doherty, John, 2010, PEST–Model-independent parameter estimation–User manual (5th ed., with slight additions): Brisbane, Australia, Watermark Numerical Computing, 336 p.

Schreüder, W. A., 2009, Running BeoPEST. *in* Proceedings of the 1st PEST Conference, Potomac, Md., 1–3 November.